# 15-827 Security and Cryptography

# Lecture 2

*Jeannette M. Wing*

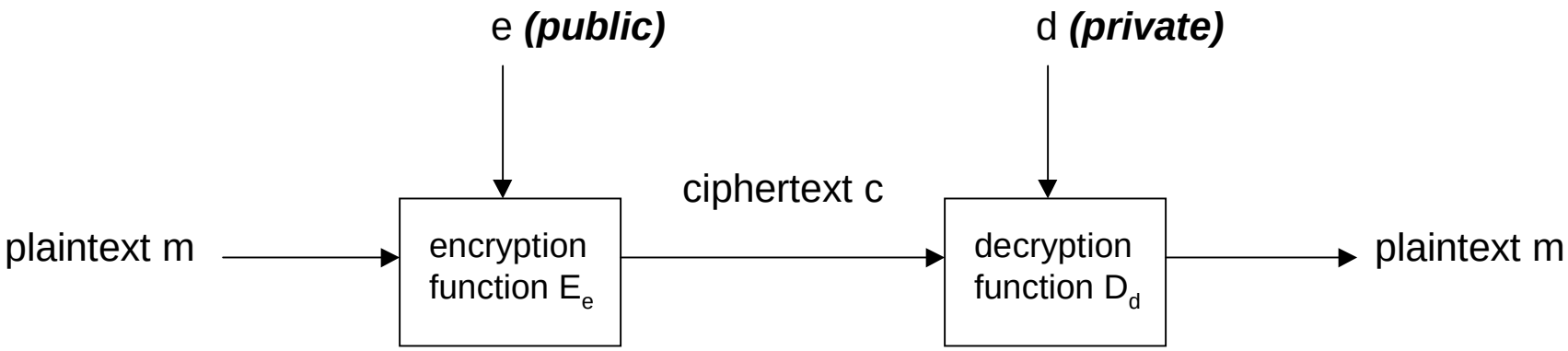
# Outline

- Public-key cryptography
  - RSA
  - Rabin
  - ElGamal
- Public-key algorithms
  - Digital signature
    - ElGamal, DSS
  - Key exchange
    - Diffie-Hellman, Station-to-Station
  - Cool things
    - Secret sharing
    - Remote coin flipping
    - Oblivious transfer
    - Blind signatures

# Cryptosystem Review

- Cryptosystem = <M, C, K, E, D>
  - M = finite set of possible *plaintexts*
  - C = finite set of possible *ciphertexts*
  - K = *key space*, finite set of possible keys
  - $\{E_e: e \in K\}$ = finite set of encryption functions, $E_e$: M -> C
  - $\{D_d: d \in K\}$ = finite set of decryption functions, $D_d$: C -> M, such that
    - For each e in K, there is a unique key d in K such that $D_d = E_e^{-1}$.
    - That is, $D_d(E_e(m)) = m$ for all m in M.

# Public-Key Cryptosystem (pictorially)



# Applications of Public-Key Cryptography

1. Exchanging messages
   - Sender encrypts m with receiver's public key.
   - Receiver decrypts with private key.
2. Key distribution
   - Alice uses PKC to send a fresh, randomly generated, session key K to Bob.
   - Alice and Bob communicate using session key K using symmetric key cryptosystem.
     - Symmetric-key crypto is about 3 orders of magnitude faster than most public-key cryptosystems.

# Applications of Public-Key Cryptography (cont'd)

3. Digital signatures
   - Reverses roles of public and private keys.
   - Alice uses private key to encrypt either
     - checksum, or
     - entire document
   - Bob (or anyone) uses Alice's public key to decrypt. Bob knows received message is from Alice.
   - Only useful if public keys are associated with individuals, handled by certificates.
4. Certificates
   - Trusted third party, *certification authority*, signs a small certificate <*name*, *public-key*$_{name}$>
   - Certificate indicates that only *name* knows the private key that goes with *public-key*$_{name}$

# Public-Key Cryptography History

- Concept invented by Diffie and Hellman and independently by Merkle in 1976.
- Many PKC algorithms published since then.
  - Many are insecure.
  - Of the “secure” ones, many are impractical (key too large, |ciphertext| >> |plaintext|, …).
  - A few are secure and practical.
    - Some good for key distribution only.
    - Some good for encryption (and hence key distribution too).
    - Some good for digital signatures only.
    - Three good for both encryption and digital signatures: RSA, Rabin, ElGamal.
    - All these are slow.

# RSA: Intro

- Based on intractability of integer factorization
  - Key idea: It is easy to find two large primes but hard to factor the product of the two primes.
- Deterministic: For each plaintext and a public key, there is only one ciphertext.
- Math needed:
  - Fermat's theorem
    - $a^{p-1} = 1 \bmod p$ where $\gcd(a, p) = 1$
  - Algorithms
    - Primality testing (e.g., Miller-Rabin; Chapter 4).
    - Extended Euclidean algorithm (Algorithm 2.107).
      - Given a and b, return d, x, and y such that $ax + by = d$ and $\gcd(a, b) = d$
    - Exponentiation (Algorithm 2.143).

# RSA: How It Works

- **Key generation**
  - Generate large primes p, q. (Primality testing)
  - Compute n = pq and $\phi(n) = (p-1)(q-1)$.
  - Choose small e, relatively prime to $\phi(n)$. (Often e = 3.)
  - Compute unique d such that

    $ed = 1 \bmod \phi(n)$ (Extended Euclidean algorithm)
  - Public key = (n, e). Private key = d.
- **Encryption**
  - Compute $c = m^e \bmod n$ (Exponentiation algorithm)
- **Decryption**
  - Compute $m = c^d \bmod n$

# RSA: Why It Works

| | |
|---|---|
| $m^{p-1} = 1 \bmod p$ | Fermat's theorem, $m \neq 0$ |
| $m^{\phi(n)} = 1 \bmod p$ | since p-1 divides $\phi(n)$ |
| $m^{k\phi(n)} = 1 \bmod p$ | for any k |
| $m^{k\phi(n)+1} = m \bmod p$ | even if m = 0 |
| $m^{ed} = m \bmod p$ | since $ed = 1 \bmod \phi(n)$, then $ed = 1 + k\phi(n)$ for some k |
| $m^{ed} = m \bmod q$ | parallel argument as above |
| $m^{ed} = m \bmod n$ | |

Hence,

$c^d \bmod n = (m^e)^d \bmod n = m \bmod n$

# The RSA Problem and Its Relation to Factoring

- RSAP
  - Given positive integer *n* that is a product of two distinct odd primes, *p* and *q*, a positive integer *e* such that *gcd(e, (p-1)(q-1)) = 1*, and an integer *c*, find integer *m* such that $m^e = c \bmod n$.
    - Adversary wants to recover *m* from *c* and *(n, e)* *(public info)*.
    - There is no efficient algorithm known for this problem.
- FACTORING
  - Given positive integer *n*, find its prime factorization

    $n = p_1^{e1} p_2^{e2} \ldots p_k^{ek}$.
- FACTORING is harder than or equal to the RSAP.
    - RSAP $\leq$ FACTORING
- It is believed that they are computationally equivalent but no proof is known.

# The RSA Problem Cont'd

- Note, if you can factor *n*, then you can compute $\phi(n)$ and thus *d*.
  - So, computing *d* must be hard too. I.e., you should not be able to compute *d* from *(n, e)* easily.
  - Fact 8.6: The problem of computing the RSA decryption exponent *d* from public key *(n, e)* and the problem of factoring *n* are computationally equivalent.

# RSA Properties

- Keys e, d can be used in either order.
  - Useful for digital signatures.
- Homomorphic property:
  - $(m\ m')^e = m^e\ m'^e = c\ c'$
  - Double-edged sword
    - Useful for Chaum's digital cash application
    - Security risk: Signing m and m' separately magically gives recipient signature for (m m')
      - Adaptive chosen-ciphertext attack (pp. 288-289)
        - » Adversary wants to decrypt $c = m^e \bmod n$
        - » Adversary conceals c and sends Alice $c' = cx^e \bmod n$ (for random x)
        - » Alice computes for Adversary $m' = (c')^d \bmod n$
        - » $m' = (c')^d = c^d\ (x^e)^d = mx \bmod n$
        - » So, Adversary computes $m = m'\ x^{-1} \bmod n$

# Attacking RSA (pp. 287-291)

- Small encryption exponent e (tradeoff between efficiency and security).
- Small decryption exponent d (should be roughly the size of n).
- Forward search attack (if message space is small or predictable, encrypt all possible plaintexts until you obtain c).
- Exploit homomorphic property.
- Common modulus attack.
- Cycling attack.
- Message concealing.
- Size of modulus.
- Selection of primes.

# Rabin Public-Key Encryption (Section 8.3)

- First public-key method that is *provably secure*.
- Based on intractability of factoring and the computational equivalence of SQROOT and FACTORING.
- SQROOT:
  - Given a composite integer *n* and a quadratic residue *a mod n*, find a square root of *a mod n*.
- Provably secure against a passive adversary but succumbs to chosen-ciphertext attack as well as other attacks similar to those used against RSA.

# Rabin: How It Works

- Key generation
  - Generate two large random primes p and q.
  - Compute n = pq.
  - Public key = n.  Private key = (p, q).
- Encryption
  - Compute and send $c = m^2 \bmod n$.
- Decryption
  - Find the four square roots $m_1, m_2, m_3, m_4$ of c mod n.
    - Relies on Chinese remainder theorem.
      - If x = y mod p and x = y mod q then x = y mod pq.
      - A number has two roots mod p, and so four roots mod pq.
  - The message is one of the four.

# Example

- p = 3, q = 5. So, n = 15. Let c = 1 mod 15.
- Four square roots of 1 mod 15

| mod 15 | mod 3 | mod 5 |
|---|---|---|
| 1 | 1 | 1 |
| 4 | 1 | -1 |
| 11 | -1 | 1 |
| 14 | -1 | -1 |

- If we take two square roots and add them, and then GCD with pq, we are likely (50%) to get p or q.

| mod 15 | mod 3 | mod 5 |
|---|---|---|
| 1 | 1 | 1 |
| 11 | -1 | 1 |
| 12 | 0 | 2 |

- Since (11+1) mod 3 is 0, gcd(12, 15) is a factor of 15 (namely, 3), we get 50% chance that 3 is p or q.
- So, if we have all the roots, we can factorize, and if we have any root we can factor probabilistically:
  - Take a random $r^2$ mod pq.
  - Compute its square root s.
  - There will be a 50% chance that gcd(r+s, pq) is p or q.
- Iterate, each time increasing the chance that we get p or q.
  - Use this constructive technique as chosen-ciphertext attack.
- Thus, finding the roots must be at least as hard as factorization. (In fact, we know they are equivalent.) This is a better guarantee than RSA.

# ElGamal: Intro

- Based on intractability of discrete logarithm problem
  - DLP: Given a prime *p*, a generator $\alpha$ of $Z_p^*$, and an element $\beta$ of $Z_p^*$, find the integer *x, 0* $\leq x \leq$ *p-2* such that $\alpha^x = \beta \ mod\ p$.
- Non-deterministic
  - One-to-many mapping of plaintext to ciphertext.
  - Based on choosing a random integer in encryption process.
- Math needed
  - Algorithms
    - Selecting a prime p and a generator for $Z_p^*$ (Algorithm 4.84).
    - Exponentiation (Algorithm 2.143).

# ElGamal: How It Works

- **Key generation**
  - Generate large, random prime p, and a generator $\alpha$ of multiplicative group $Z_p^*$.
  - Choose random integer a such that $1 \leq a \leq p-2$ and compute $\alpha^a \bmod p$.
  - Public key = $(p, \alpha, \alpha^a)$. Private key = a.
- **Encryption**
  - Choose random k such that $1 \leq k \leq p-2$.
  - Compute $\gamma = \alpha^k \bmod p$ and $\delta = m (\alpha^a)^k \bmod p$.
  - Send $c = (\gamma, \delta)$.
- **Decryption**
  - Compute $\gamma^{p-1-a} \bmod p$. Note $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$.
  - Compute $m = \gamma^{-a} \delta \bmod p$.

# ElGamal: Why It Works

$$\gamma^{-a}\,\delta = \alpha^{-ak}\, m\, (\alpha^{a})^{k} = m$$

# Example

- p = 2579, $\alpha$ = 2, a = 765, and hence

  $$\alpha^{a} = 2^{765} \bmod 2579 = 949$$

- Suppose Alice wants to send message m = 1299 to Bob.
- Alice
  - Chooses random k = 853.
  - Computes

    $\gamma = 2^{853} \bmod 2579 = 435$ and

    $\delta = 1299 \times 949^{853} \bmod 2579 = 2396$.
  - Sends Bob (435, 2396).
- Bob, using public key (2579, 2, 949) and private key 765,
  - Computes $2396 \times (435^{765})^{-1} \bmod 2579 = 1299$.

# ElGamal Notes

- Efficiency
  - Message expansion: ciphertext is twice as long as plaintext.
  - Encryption requires two modular exponentiations.
- Randomization in encryption
  - Decreases effectiveness of chosen-plaintext attack
- Hardness
  - Security is based on discrete logarithm problem in $Z_p^*$ but the equivalence has not been proven.
  - Is equivalent to Diffie-Hellman problem (see Section 3.7).
  - Is critical that different random integers k be used to encrypt different messages.
- Generalized ElGamal
  - Instead of multiplicative group $Z_p^*$, work in any finite cyclic group G (q.v. elliptic curves).

# ElGamal Digital Signature: How It Works

- Key generation is the same
  - Public key = $(p, \alpha, \alpha^a)$. Private key = a.
- Signature generation
  - Choose random secret integer k, $1 \leq k \leq p-2$, $\gcd(k, p-1) = 1$.
  - Compute $r = \alpha^k \bmod p$, $k^{-1} \bmod (p-1)$, $s = k^{-1}(h(m) - ar) \bmod (p-1)$.
  - Signature for m is (r, s).
- Signature verification
  - Verify that $1 \leq r \leq p-1$; if not, reject.
  - Compute $v_1 = (\alpha^a)^r r^s \bmod p$, h(m), and $v_2 = \alpha^{h(m)} \bmod p$.
  - Accept signature iff $v_1 = v_2$.

# ElGamal Digital Signature: Why It Works

$$s = k^{-1} (h(m) - ar) \bmod (p-1)$$

$$ks = (h(m) - ar) \bmod (p-1)$$

$$h(m) = (ar + ks) \bmod (p-1)$$

=>

$$\alpha^{h(m)} = \alpha^{ar + ks} = (\alpha^{a})^{r} r^{s}$$

$$v_2 = v_1$$

# Digital Signature Standard (DSA ElGamal)

- DSS is a modification of the ElGamal Signature Scheme.
- Proposed in 1991, adopted in 1994 as a standard.
  - Controversy because selection process by NIST was not public.
- Good for signing only, not encryption.
- Major differences:
  - Use + instead of - in definition of s in signature generation.
  - Change verification condition accordingly to

    $\alpha^{k} = (\alpha^{h(m)/s} (\alpha^{a})^{r/s} \bmod p) \bmod q$
  - Choosing p and q (which are both part of the public key):
    - q is a prime such that $2^{159} < q < 2^{160}$.
    - p is a prime such that q divides (p-1).
    - Innovative Idea:160-bit messages are signed using a 320-bit signature, but the computations are done using a 512-bit modulus p. I.e., work in a subgroup of size $2^{160}$.

# Elliptic Curves

- The problem with integers is that there is only one set of them! We'd like more choices.
- Elliptic curves over the real numbers are curves of the form:

$$y^2 = x^3 + ax + b$$

- and cannot be factorizable:

$$4a^3 + 27b^2 \neq 0$$

- They can be used to define an abelian group:
  - Let x and y be elements of some field F.
  - If (x, y) satisfies the equation, then it's an element of the group.
  - Add special point O called the point at infinity.
- Do not have certain "smoothness" properties, presumably making it hard for cryptanalysis.

# Addition

- Geometric interpretation
  - Given two points on the curve, draw a line between them.
  - It's a cubic equation so there will be a third intercept.
  - Take the intercept and reflect it over the x-axis.
  - The resulting point is the sum of the two original points.
- Doubling: Use the tangent to the curve.
- Additive identity is the point O with infinite y value.
- Not obvious that addition is associative.

# Algebraic Definition

- Addition ($P \neq Q$)

  $P + Q = R$

  $s = (y_p - y_q)/(x_p - x_q)$

  $x_r = s^2 - x_p - x_q$

  $y_r = -y_p + s(x_p - x_r)$

  Doubling ($P = Q$)

  $2P = R$

  $s = (3x_p^2 + a)/2y_p$

  $x_r = s^2 - 2x_p$

  $y_r = -y_p + s(x_p - x_r)$

- Multiplication by repeated addition
  - 26 = 11010 binary
  - $26P = 2(2(2(2(0+P)+P)+0)+P)+0$

# Elliptic Curve Cryptography

- Elliptic curves over $Z_p$ for prime $p > 3$.
  - Modular arithmetic, but similar to interpretation over reals.
- Key idea: Define multiplication by repeated addition.
  - Application to discrete logarithm problem
    - Let $Q = kP$. Given P and Q, find k.
    - E.g., for Diffie-Hellman key exchange, publish curve E and point P on curve. Secret key is x-coordinate of point $r_A r_B P$ (for secret random numbers $r_A$, $r_B$ chosen by A and B).
- Pros and cons
  - It takes more basic field operations, but each field operation is smaller (performance win).
  - Keys are smaller.
  - Shortest unpadded message is smaller.
- http://www.certicom.com/ecc/

# Homework Exercise

- Show how to use elliptic curves to implement ElGamal's encryption/decryption scheme.

# Other Public-Key Algorithms

- *Digital Signatures*
  - *DSS (ElGamal)*
- Key exchange
  - Diffie-Hellman
  - Station-to-Station
- Cool things
  - Secret-sharing
    - Key escrow
  - Remote coin flipping
  - Oblivious transfer
  - Blind signatures
  - Zero-knowledge proofs, zero-knowledge identification (Lecture 4)

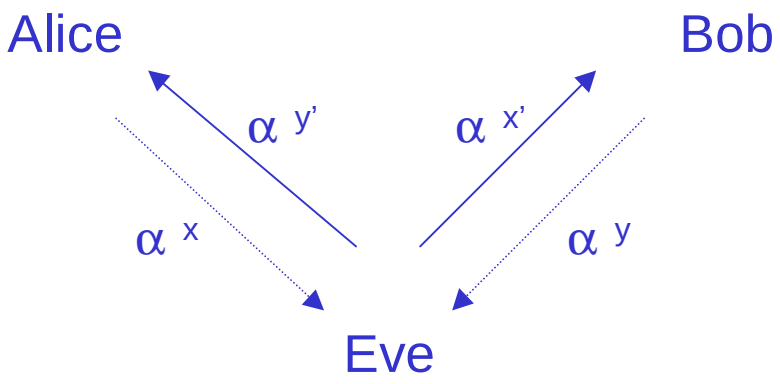
# Diffie-Hellman Key Exchange

- First practical solution to key distribution problem.
- Based on difficulty of Diffie-Hellman problem and DLP.
- Provides protection against passive, but not active, adversaries.
- Here it is:
  - One-time setup
    - Choose and publish prime p and generator $\alpha$ of $Z_p^*$, $2 \leq \alpha \leq p-2$
  - Message exchange

    A -> B: $\alpha^x \bmod p$ for secret random x that A chooses, $1 \leq x \leq p-2$

    B -> A: $\alpha^y \bmod p$ for secret random y that B chooses, $1 \leq y \leq p-2$
  - A and B share secret key k.
    - B receives $\alpha^x$ and computes $k = (\alpha^x)^y \bmod p$.
    - A receives $\alpha^y$ and computes $k = (\alpha^y)^x \bmod p$.

# Man-in-the-Middle Attack (Active Adversary)



# Station-to-Station (STS) Protocol

- Due to Diffie, Oorschot, and Wiener.
- One-time setup: Choose p and $\alpha$ as before. Use RSA public and private keys for signatures. Assume A has access to B's public key and v.v.

1. A -> B: $\alpha^x \bmod p$ for random x mod p-1 chosen by Alice
2. B -> A: $\alpha^y \bmod p$, $E_k(S_B(\alpha^y, \alpha^x))$ for random y mod p-1 chosen by Bob; k is session key.
3. A -> B: $E_k(S_A(\alpha^x, \alpha^y))$

- After A receives Message 2 from B, she is assured that B has $\alpha^x$ and that the $\alpha^y$ she received is from Bob since it contains B's signature. A sends B a confirmation in Message 3, so ditto for B after he receives Message 3 from A.
- A and B both compute $k = \alpha^{xy} \bmod p$ and is the key used in encryption and decryption above.
- STS eliminates Man-in-the-Middle attack.

# Secret Sharing

- Motivation
  - Two out of three of the President, the Defense Minister, and the Defense Ministry, are needed to control access to nuclear weapons in Russia [Time, 1992].
  - Compute the average CSD faculty salary without revealing any one individual's salary.
- Solution
  - Secret sharing
    - Shamir's (t, n) threshold scheme ($t \leq n$).

# Shamir's (t, n) Threshold Scheme

- Based on
  - Fact that polynomial $y = f(x)$ of degree t-1 is uniquely defined by t points $(x_i, y_i)$ with distinct $x_i$.
- Method
  - Choose prime $p > \max(S, n)$. (All math done mod p.)
  - Define $a_0 = S$ (secret).
  - Choose t-1 random, independent coefficients $a_1, \ldots, a_{t-1}$ and form polynomial:

$$f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{t-1}x^{t-1}$$

  - Trusted party computes and distributes secret shares $S_i = f(i)$, $1 \leq i \leq n$, along with public index i.
  - Any group of t or more users pool their shares and compute coefficients $a_j$, $1 \leq j \leq t-1$ of f(x), e.g., by Lagrangian interpolation. They can then compute $f(0) = a_0 = S$.
    - Any group of t-1 or less users has no information (perfect secrecy).

# Example

- Suppose p = 17, t = 3, w = 5; $x_i$ = i $1 \leq i \leq 5$.
- Suppose users 1, 3, and 5 pool their shares, which are 8, 10, and 11, respectively.
- Polynomial is $f(x) = a_0 + a_1x + a_2x^2$
- Compute f(1), f(3), and f(5) in $Z_{17:}$
  - $a_0 + a_1 + a_2 = 8$
  - $a_0 + 3a_1 + 9a_2 = 10$
  - $a_0 + 5a_1 + 8a_2 = 11$
- Unique solution in $Z_{17}$: $a_0$ =13, $a_1$ = 10, $a_2$ = 2. S = 13.

# Key Escrow

- Goal: Provide encryption of user traffic (voice, data) such that session keys are available to properly authorized third parties under special circumstances ("emergency access").
- Proposal: Divide key into n pieces and give each to n individuals so that any k of n can decrypt but no (k-1) can.
- Clipper/Capstone/Skipjack/MOSAIC/Tessera/Fortezza/EES:
  - Original gov't proposal:
    - For Clipper, k = n = 2, and gov't was one of the 2. All done in hardware. Idea: force crypto devices to record key in encrypted format. Controversy because
      - uses then secret algorithm, Skipjack, only made public this summer.
      - if gov't's aim is to deter criminals, they will surely find a way around it.
  - Current gov't proposal: do it in software.
- Private companies are proposing similar technologies, e.g., Intertrust's "rights management."

# Remote Coin Flipping

- Suppose Alice and Bob want to flip a coin over the telephone. Alice flips a coin. Bob says “Heads.” How can Bob trust Alice to know he won or lost?
- Many techniques, e.g., using square roots, exponentiation mod p, Blum integers.

# Example

Multiples of 4: 0 4 8

Odds: 1 3 5 7 9

Leftovers: 2 6

- Fact: Every odd integer is either one greater than or one less than a multiple of 4.
  - Represent Type +1 integers as 4a +1.
  - Represent Type -1 integers as 4a -1.
  - Product of two Type +1 integers is Type +1.

    (4a+1)(4b+1) = 16ab + 4a + 4b + 1 = 4(4ab+a+b) + 1
  - Product of two Type -1 integers is Type +1.

    (4a-1)(4b-1) = 16ab - 4a - 4b + 1 = 4(4ab-a-b) + 1
- Hence, looking at product, you can't tell which type the factors were.

# Remote Coin Flip Protocol

1. Alice randomly picks one of the two types. (Flips coin.)
   She multiplies together two randomly chosen large primes of that type and sends the product to Bob.
2. Bob guesses at the type of the factors. (Says “heads” or “tails”.)
3. Alice reveals the prime factors so Bob can verify whether he won (guessed correctly) or lost.

This method relies on hardness of factoring, and the fact that there are an infinite number of both types of primes.

# Blum Integers

Def: A *Blum integer* is a composite integer of the form $n = pq$, where $p$ and $q$ are distinct primes each congruent to 3 mod 4.

Fact 2.157: Let $n = pq$ be a Blum integer, and let $a$ in $Q_n$ (set of quadratic residues mod n). Then $a$ has precisely four square roots *mod n*, exactly one of which is also in $Q_n$; that one is called the principal square root of *a mod n*.

Example: For Blum integer n = 21. The four square roots of a = 4 are 2, 5, 16, and 19, of which 16 is also in $Q_{21}$ (= {1, 4, 16}).

Def: Let $a$ in $Z_n^*$. $a$ is a *quadratic residue mod n* if there exists an $x$ in $Z_n^*$ such that $x^2 = a \bmod n$.

# Homework Exercise

- Use Blum integers to solve the remote coin flipping problem.

# Oblivious Transfer

- Suppose Alice wants to send a message to Bob such that:
  - Bob has the probability of 1/2 of actually receiving the message.
  - Alice does not know if Bob received the message but Bob does.
- Not done in practice, but it's a building block.
  - Useful for simultaneous exchange of secrets, contracts, signatures, and certified mail.
- Many types of oblivious transfer, but all equivalent:
  - I have two secrets and you get one.
  - I have n secrets and you get one.
  - I have one secret and you get it with probability 1/2.
  - Generalizable to other probabilities of success.
- Solutions due to Rabin, Blum, Even, ...

# Rabin's Solution

1. Alice sends Bob the product of two primes: $n = pq$.
2. Bob chooses a random $x < n$, such that gcd(x, n) = 1. He sends Alice:

$$a = x^2 \bmod n$$

3. Alice, knowing $p$ and $q$, computes the four roots of $a$: $x$, $n-x$, $y$, and $n-y$. She chooses one at random and sends it to Bob.
4. If Bob receives $y$ or $n-y$, he can compute the $gcd(x+y, n)$, which is either $p$ or $q$. Then, of course, $n/p = q$.

   If Bob receives $x$ or $n-x$, he can't compute anything.

# Another Solution Using ElGamal (so you get the general idea)

1. Let p be a large prime, and $\alpha$ be a generator mod p. (Finding discrete logs mod p should be hard.)
2. Assume that it is infeasible for anyone to find the discrete log of 2 (i.e., to find the x such that $2 = \alpha^x \bmod p$).
3. Bob publishes two numbers, v and v' = 2v. (Because of previous point, he knows the discrete log of exactly one of them.)
4. If Alice wants to send m to Bob, Alice picks v or v', and encrypts m with it, for example using ElGamal. E.g., say that Alice picked v, then

   $(\gamma, \delta) = (\alpha^k, m\, v^k)$
5. If Alice picks the correct one (which Bob knows the log of), then Bob can read the message. Otherwise, he can't.

# Blind Signatures

- Problem: Alice wants Bob to sign message m blindly, i.e., make him sign something without his seeing what it is he is signing.
- One solution (due to Chaum), using RSA.
  - Bob has: public key = e; private key = d; public modulus = n.

  1. Alice chooses random value, k, $1 \leq k \leq n-1$, gcd(n, k) = 1. (k is called the blinding factor.) She blinds m by computing
     $t = mk^e \bmod n$
  2. Bob signs t
     $t^d = (mk^e)^d \bmod n$
  3. Alice unblinds $t^d$ by computing (she divides out the blinding factor)
     $s = t^d/k \bmod n$
  4. The result is
     $s = m^d \bmod n$
- Why it works

  $t^d = (mk^e)^d = m^dk \bmod n$, so $t^d/k = m^dk/k = m^d \bmod n$

# Next lecture

- Topics: Authentication
  - Zero-knowledge proofs, zero-knowledge identification
  - Authentication and key establishment protocols
- Readings
  - For this lecture
    - Sections 8.1-8.4, 11.1-11.3.4, 11.5.1-11.5.2, 12.7, Jurisic and Menezes paper. Section 13.4 and 13.8.3.
    - Background: 9.1, 9.2, 9.4 (on MD4, MD5, and SHA-1 especially).
    - Edification: 11.4, 11.5.3, 11.8, rest of Chapter 9, Kocher paper.
  - For next lecture
    - Chapters 10, 12, Quisquater, Guillou, and Berson paper.